APLIKASI GENERATOR AKORD DENGAN MENGGUNAKAN FONT *NOTANGKA.TTF* DAN MENGADAPTASI LOGIKA *DIRECT PRODUCT* PADA NOTASI MUSIKAL ANGKA

Eko Sediyono, Yessica M. Susanto, Theopillus Herman W.
Fakultas Teknologi Informasi, Universitas Kristen Satya Wacana
Jl. Diponegoro 52-60, Salatiga 50711, Indonesia
Email: mar1_0nette@yahoo.com, ekose1@yahoo.com, erman_wellem@yahoo.com

ABSTRAK: Paper ini membahas perancangan aplikasi pembangkit akord dari not angka yang ditulis dengan editor teks dengan font *NotAngka2.ttf*. Langkah-langkah yang digunakan untuk membangkitkan akord menggunakan teknik kompilasi. Tahap pertama adalah *lexical analysis* yang menyaring *string* dan membentuk *token*. Tahap ke dua adalah *syntax analysis* dimana input yang masuk diteliti struktur gramarnya. Selanjutnya dengan menggunakan logika perkalian langsung (*direct product logic*), maka dapat dibentuk akord yang sesuai dengan lagu masukannya. Aplikasi yang dibangun masih dalam tahap awal, sehingga inputnya masih terbatas pada lagu yang sangat sederhana, yang dibatasi pada patokan nada dasar, ketukan, jenis akord, serta tangga nada yang digunakan

Kata kunci: *akord, direct product, syntax analysis, lexical analysis*

***ABSTRACT**: Information technology has brought a big influence in other fields of science, especially in music. One part of the musical arrangement process is chord defining. Direct product is one of vector multiplication types. Although its using in vector, but in music, it can be used for searching all possibilities of chord arrangement. This chord generator will use NotAngka2.ttf as a basic for input writing. Because of the unexistence of application for writing song in number notation with NotAngka2 font type and generating some chord arrangements for the song, so this topic become a focus for application designing and implementing.*

***Keywords:** chord, direct product, notangka2.ttf, number notation*

PENDAHULUAN

Di dalam musik, aransemen adalah menulis kembali sebuah lagu dengan menambahkan materi baru di dalamnya atau memperluas sketsa komposisi. Jika dalam mengadaptasi musik tidak disertai dengan material baru, maka tidak termasuk aransemen tetapi transkripsi atau orkestrasi [1]. Persatuan Seniman Musik Amerika mendefinisikan aransemen sebagai seni mempersiapkan dan mengadaptasi komposisi musik yang telah ditulis untuk dipresentasikan secara lebih daripada aslinya.

Dengan kemajuan teknologi komputer dan perangkat lunaknya, seorang awam yang tidak dapat memainkan satu alat musik pun dapat menggubah lagu lengkap dengan aransemenya. Sebagai contoh pengguna awam dapat menggunakan perangkat lunak *Guitar Pro* untuk menggubah sebuah lagu dengan alat musik Gitar. *CakeWalkPro* adalah sebuah perangkat lunak untuk membantu mengaransir lagu dari sebuah alat musik menjadi komposisi orkestra.

Software tersebut mensyaratkan penggunanya mengerti notasi balok, padahal orang awam kebanyakan hanya mengerti notasi angka, dan buku-buku lagu untuk umum biasanya ditulis dengan notasi angka.

Paper ini membahas perancangan aplikasi yang dapat digunakan untuk menambahkan akord pada sebuah lagu dengan notasi angka yang telah ditulis dengan menggunakan pengolah kata yang dapat memanfaatkan *font* dalam *file NotAngka2.ttf*. *File* tersebut bersifat *opensource* dan dapat diunduh di *http://www.im-mc.org/download.php? view.23.*

Untuk menganalisis masukan dari pengolah kata dengan font yang telah ditentukan tersebut, dalam paper ini digunakan langkah-langkah analisis seperti dalam teknik kompilasi. Tahap pertama adalah *lexical analysis* yang menyaring *string* dan membentuk *token*. Tahap kedua adalah *syntax analysis* dimana input yang masuk diteliti struktur *grammar*nya. Selanjutnya dengan menggunakan logika perkalian langsung (*direct product logic*), maka dapat dibentuk akord yang sesuai dengan lagu masukannya.

KAJIAN PUSTAKA

Beberapa penelitian yang terkait dengan penyusunan akord dari suatu lagu telah dilakukan. Takuya Yoshioka meneliti tentang pengenalan karya musik dari sinyal suara pada rekaman *compact-disc* dengan menggunakan batasan frekuensi akord [2]. Pengenalan akord musik yang otomatis ini biasanya

digunakan untuk sistem pencarian musik. Penelitian ini menyebutkan bahwa tidak ada satupun metode yang dapat mengenali akord musik dari sinyal *audio* yang kompleks terutama yang mengandung suara drum dan suara vokal manusia. Pencarian dengan metode ini hanya menghasilkan kurang lebih 77% keberhasilan.

Paiement menjelaskan tentang model progresi akord yang memakai probabilitas [3]. Dengan input berupa suara, diperkirakan subtitusi antar akord agar dapat dihasilkan grafik progresi akord yang lebih halus.

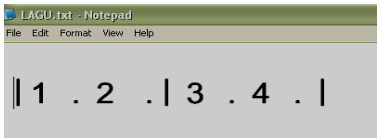
Sementara untuk aplikasi mengenai akord juga dibuat oleh Chemillier yang mengimplementasikan generator akord untuk musik *DJ* yang didasarkan pada aturan-aturan harmoni *jazz* [4]. Sistem yang dibuat mampu menggunakan *generator* akord berbasis gramar. Gramar yang dipakai pada aplikasi ini adalah gramar *Steedman* yang menggambarkan subtitusi harmoni yang dimainkan oleh musisi *jazz*. Sementara itu, dari kajian pustaka yang telah dilakukan belum ada *generator* akord dengan masukan teks dengan huruf *NotAngka2.ttf* .

PERANCANGAN SISTEM

Metode yang digunakan untuk menganalisis dan merancang sistem *generator* akord ini adalah metode *waterfall* 5. Tahap-tahapnya adalah sebagai berikut.

Analisis Kebutuhan

Input berupa notasi angka dalam format .doc dengan font *NotAngka2.ttf*. Aplikasi akan menampilkan pesan kesalahan jika teks yang dimasukkan tidak valid baik secara karakter maupun secara struktur yang telah ditentukan. Contoh teks input yang sesuai dengan struktur dapat dilihat pada Gambar 1.



Gambar 1. *Input* yang akan diproses

Jenis karakter yang dapat dikenali oleh system yang dibuat seperti pada Tabel 1.

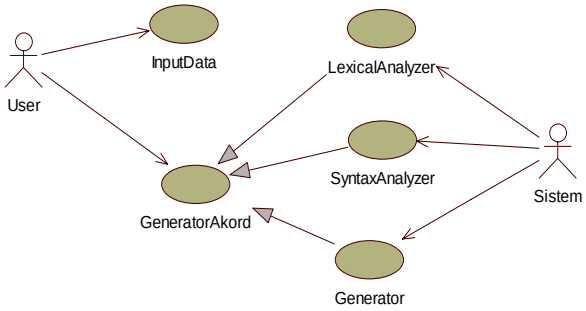
Tabel 1. Tabel Jenis Karakter *Input* yang diperbolehkan

Jenis Input	Karakter
Notasi angka	1, 2, 3, 4, 5, 6, 7
Tanda henti	0
Titik	.
Garis birama	|
Garis refrein	{
Garis tutup birama	}
Spasi	spasi
Ganti baris/enter	\n
Setengah ketuk	_
Seperempat ketuk	=

Perancangan

Rancangan mengenai sistem yang dibuat dibagai menjadi 2 bagian utama, yaitu *Input file* dan *Generator* akord. *Use-case* Generator Akord dibagi menjadi 3, yaitu *use-case Lexical Analyzer*, *use-case Syntax Analyzer*, dan *use-case Generator*. *Use-case diagram* rancangan sistem ini digambarkan pada Gambar 2. Pada *use-case Input Data*, pengguna memasukkan data berupa notasi angka ke dalam *file*. *File* yang dimasukkan dibaca dan diubah ke dalam bentuk yang dapat diolah untuk proses selanjutnya.

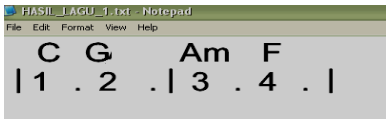
Pada *use-case Lexical Analyzer* dan *Syntax Analyzer* terjadi proses pengecekan *input* dari kesalahan, baik berupa penyaringan karakter *input* yang masuk maupun pengecekan sesuai aturan sintaks yang berlaku. Sebelum *input* diproses oleh *Generator*, *input* harus bebas dari kesalahan. Untuk itu diperlukan kinerja dari *use-case Lexical Analyzer* dan *Syntax Analyzer* ini.



Gambar 2. *Use-case diagram* Sistem

Pada *InputData* terjadi proses dimana pengguna memasukkan *input* berupa rangkaian notasi angka ke dalam sebuah *file* dan kemudian *file* tersebut siap untuk diproses lebih lanjut.

Bentuk *Output*



Gambar 3. *Output* setelah diproses

Gambar 3 memperlihatkan bentuk *output* dari sistem yang dirancang yaitu akord tampil diatas notasi angka yang telah dimasukkan ke dalam sistem.

LEXICAL ANALYZER

Dalam l*exical analyzer*, terdapat pengecekan *input* yang masuk dan memecahnya menjadi *token*. Dalam rancangan aplikasi *generator* akord, *token* yang dihasilkan memiliki bentuk seperti pada Tabel 2.

Tabel 2. Jenis *token lexical analyzer*

Jenis Token Lexical Analyzer
=not
=titik
=nol
-not
-titik
-nol
not
titik
nol
garis_reff
garis_tutup
garis_birama

SYNTAX ANALYZER

a. Aturan penulisan 1 ketuk yang berisi 4 buah notasi ¼ ketuk diberi simbol ***not4*** tercantum pada Tabel 3.
b. Aturan penulisan 1 ketuk yang berisi 1 notasi ½ ketuk dan 2 buah notasi ¼ ketuk diberi simbol not12 tercantum pada Tabel 4.

Aturan produksi dari input dengan notasi angka ditulis dalam bentuk *Backus Naus Form* (BNF). *Backus Naur Form* merupakan bentuk formal yang menggambarkan aturan sintaks yang diberikan oleh sebuah bahasa. *Backus Naur Form* ini dibuat oleh John Backus dan Peter Naur.

<lagu> : := <ketuk-awal> <1ketuk> <1ketuk> <1ketuk> <garis-birama> <tengah> <1ketuk> <1ketuk> <1ketuk> <1ketuk> <garis-tutup> | <ketuk-awal> <1ketuk> <1ketuk> <1ketuk> <garis-birama> <1ketuk> <1ketuk> <1ketuk> <1ketuk> <garis-tutup> | <ketuk-awal> <1ketuk> <1ketuk> <garis-birama> <tengah> <1ketuk> <garis-tutup> | <ketuk-awal> <1ketuk> <1ketuk> <garis-birama> <1ketuk> <garis-tutup> | <ketuk-awal> <1ketuk> <garis-birama> <tengah> <1ketuk> <1ketuk> <garis-tutup> | <ketuk-awal> <1ketuk> <garis-birama> <1ketuk> <1ketuk> <garis-tutup> | <ketuk-awal> <garis-birama> <tengah> <1ketuk> <1ketuk> <1ketuk> <garis-tutup><ketuk-awal> <garis-birama> <1ketuk> <1ketuk> <1ketuk> <garis-tutup>

<ketuk-awal> : := <half> <nol> <half> <not> | <half> <not> <half> <not> | <half> <not> <half> <nol> | <half> <nol> <quarter> <not> <quarter> <not> | <half> <nol> <quarter> <not> <quarter> <nol> | <half> <nol> <quarter> <nol> <quarter> <not> | <half> <not> <quarter> <not> <quarter> <not> | <half> <not> <quarter> <titik> <quarter> <not> | <half> <not> <quarter> <nol> <quarter> <not> | <half> <not> <quarter> <not> <quarter> <nol> | <half> <not> <quarter> <titik> <quarter> <nol> | <quarter> <nol> <quarter> <not> <half> <nol> | <quarter> <nol> <quarter> <not> <half> <titik> | <quarter> <nol> <quarter> <not> <half> <not> | <quarter> <not> <quarter> <nol> <half> <nol> | <quarter> <not> <quarter> <nol> <half> <not> | <quarter> <not> <quarter> <not> <half> <nol> | <quarter> <not> <quarter> <not> <half> <titik> | <quarter> <not> <quarter> <not> <half> <not> | <quarter> <nol> <quarter> <not> <quarter> <not> <quarter> <not> | <quarter> <nol> <quarter> <not> <quarter> <titik> <quarter> <nol> | <quarter> <nol> <quarter> <not> <quarter> <nol> <quarter> <not> | <quarter> <nol> <quarter> <not> <quarter> <not> <quarter> <nol> | <quarter> <nol> <quarter> <not> <quarter> <titik> <quarter> <nol> | <quarter> <not> <quarter> <nol> <quarter> <not> | <quarter> <not> <quarter> <nol> <quarter> <not> <quarter> <not> <quarter> <nol> | <quarter> <not> <quarter> <not> <quarter> <not> <quarter> <nol> | <quarter> <not> <quarter> <not> <quarter> <not> <quarter> <titik> <quarter> <not> | <quarter> <not> <quarter> <not> <quarter> <not> <quarter> <not> <quarter> <not> | <not> | <nol>

<1ketuk> : := <not4> | <not12> | <not21> | <not11> | <not> | <nol> | <titik>

<garis-birama> : := **" | "**

<tengah> : := <1ketuk> <1ketuk> <1ketuk> <1ketuk> <garis-birama> <tengah> | <1ketuk> <1ketuk> <1ketuk> <1ketuk> <garis-birama> | <1ketuk> <1ketuk> <1ketuk> <1ketuk> <garis-reff> <1ketuk> <1ketuk> <1ketuk> <1ketuk> <garis-birama> <tengah> | <1ketuk> <1ketuk> <1ketuk> <1ketuk> <garis-reff> <1ketuk> <1ketuk> <1ketuk> <1ketuk> <garis-birama> | <1ketuk> <1ketuk> <1ketuk> <garis-reff> <1ketuk> <garis-birama> <tengah> | <1ketuk> <1ketuk> <1ketuk> <garis-reff> <1ketuk> <garis-birama> | <1ketuk>

<1ketuk> <garis-reff> <1ketuk> <1ketuk> <garis-birama> <tengah> | <1ketuk> <1ketuk> <garis-reff> <1ketuk> <1ketuk> <garis-birama> | <1ketuk> <garis-reff> <1ketuk> <1ketuk> <1ketuk> <garis-birama> <tengah> | <1ketuk> <garis-reff> <1ketuk> <1ketuk> <1ketuk> <garis-birama>

<garis-tutup>: := **" } "**
<quarter> : := **" = "**
<half> : := **" - "**
<titik> : := **" . "**
<nol> : := **"0"**

<not> : := <number> | **"1"** <up-down> | **"1"** <slash-up> | **"1"** <up-down> <slash-up> | **"1"** <slash-up> <up-down> | **"2"** <up-down> | **"2"** <slash-up> | **"2"** <slash-down> | **"2"** <up-down> <slash-up> | **"2"** <up-down> <slash-down> | **"2"** <slash-up> <up-down> | **"2"** <slash-down> <up-down> | **"3"** <up-down> |"**3"** <slash-down> | **"3"** <up-down> <slash-down> | **"3"** <slash-down> <up-down> | **"4"** <up-down> | **"4"** <slash-up> | **"4"** <up-down> <slash-up> |

<not4> : := <quarter> <not> <quarter> <not> <quarter> <not> <quarter> <not> |
<quarter> <titik> <quarter> <not> <quarter> <not> <quarter> <not> |
<quarter> <not> <quarter> <not> <quarter> <titik> <quarter> <not> |
<quarter> <titik> <quarter> <not> <quarter> <titik> <quarter> <not> |
<quarter> <nol> <quarter> <not> <quarter> <not> <quarter> <not> |
<quarter> <not> <quarter> <nol> <quarter> <not> <quarter> <not> |
<quarter> <not> <quarter> <not> <quarter> <nol> <quarter> <not> |
<quarter> <not> <quarter> <not> <quarter> <not> <quarter> <nol> |
<quarter> <nol> <quarter> <not> <quarter> <titik> <quarter> <nol> |
<quarter> <titik> <quarter> <nol> <quarter> <not> <quarter> <not> |
<quarter> <titik> <quarter> <not> <quarter> <nol> <quarter> <not> |
<quarter> <titik> <quarter> <not> <quarter> <not> <quarter> <nol> |
<quarter> <titik> <quarter> <not> <quarter> <titik> <quarter> <nol> |
<quarter> <nol> <quarter> <not> <quarter> <nol> <quarter> <not> |
<quarter> <not> <quarter> <nol> <quarter> <nol> <quarter> <not> |
<quarter> <not> <quarter> <nol> <quarter> <not> <quarter> <nol> |
<quarter> <nol> <quarter> <not> <quarter> <not> <quarter> <nol> |
<quarter> <titik> <quarter> <nol> <quarter> <nol> <quarter> <not> |
<quarter> <titik> <quarter> <nol> <quarter> <not> <quarter> <nol> |
<quarter> <nol> <quarter> <not> <quarter> <titik> <quarter> <nol>
… dan seterusnya

<garis-reff> : := **" { "**
<number>: := **"1"|"2"| "3" | "4"| "5"|"6"|"7"**
<slash-up> : := **" / "**
<slash-down>: := **" \ "**
<up-down> : := **" >"** | **" < "**

PENGADAPTASIAN & PENERAPAN DIRECT PRODUCT

Misalkan jika terdapat notasi angka seperti pada contoh berikut:

1 3 5 | 1 7 1 . | 1 }

maka pada proses untuk mendapatkan akord, akan dihasilkan kumpulan akord untuk masing-masing notasi pada ketukan pertama dan ketiga. Kumpulan akord pada 1 notasi dianggap sebagai 1 vektor.

C F C
Em
C
Fm
1 3 5 | 1 7 1 . | 1 }

Misalkan vektor u = (C)
v = (F, Em, C, Fm)
w = (C)

Jika logika *direct product* diadaptasikan untuk penghasilan akord maka proses dan hasilnya adalah sebagai berikut:

uvw = (uv) w
= ((C) (F, Em, C, Fm)) (C)
= (CF + CEm + CC + CFm) (C)
= (CFC + CEmC + CCC + CFmC)

Tabel 3. Aturan penulisan *not4*

Jumlah	Nol (0)	Angka	Titik (.)	Contoh NotAngka2	Contoh Arial	Aturan Token
4	0	4	0	==== **1111**	==== **1111**	=not =not =not =not
4	0	3	1	==== **.111**	==== **.111**	=titik =not =not =not
				==== **11.1**	==== **11.1**	=not =not =titik =not
4	0	2	2	==== **.1.1**	==== **.1.1**	=titik =not =titik =not
4	1	3	0	==== **0111**	==== **0111**	=nol =not =not =not
				==== **1011**	==== **1011**	=not =nol =not =not
				==== **1101**	==== **1101**	=not =not =nol =not
				==== **1110**	==== **1110**	=not =not =not =nol
4	1	2	1	==== **01.1**	==== **01.1**	=nol =not =titik =not
				==== **.011**	==== **.011**	=titik =nol =not =not
				==== **.101**	==== **.101**	=titik =not =nol =not
				==== **.110**	==== **.110**	=titik =not =not =nol
				==== **01.0**	==== **01.0**	=nol =not =titik =nol

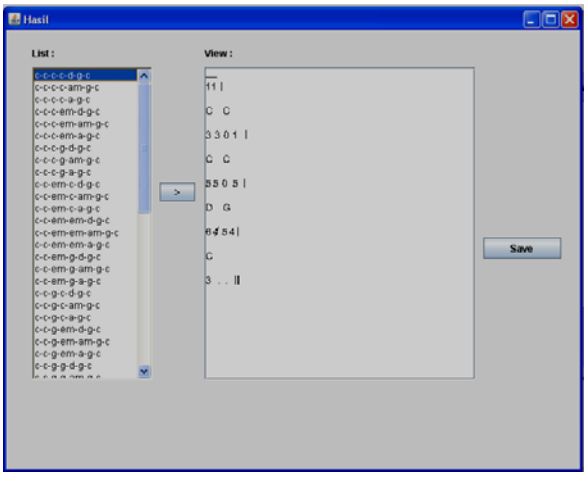
Tabel 4. Aturan penulisan not12

Jumlah	Nol (0)	Angka	Titik (.)	Contoh NotAngka2	Contoh Arial	Aturan Token
3	0	3	0	_== **111**	_== **111**	-not =not =not
3	0	2	1	_== **.11**	_== **.11**	-titik =not =not
				_== **1.1**	_== **1.1**	-not =titik =not
3	0	1	2	_== **..1**	_== **..1**	-titik =titik =not
3	1	2	0	_== **011**	_== **011**	-nol =not =not
				_== **101**	_== **101**	-not =nol =not
				_== **110**	_== **110**	-not =not =nol
3	1	1	1	_== **.10**	_== **.10**	-titik =not =nol
				_== **.01**	_== **.01**	-titik =nol =not
				_== **1.0**	_== **1.0**	-not =titik =nol
3	1	0	2	_== **..0**	_== **..0**	-titik =titik =nol
3	2	1	0	_== **010**	_== **010**	-nol =not =nol
				_== **001**	_== **001**	-nol =nol =not

Sehingga jika dilihat dari notasi maka akan menjadi seperti berikut:

C F C
C Em C
C C C
C Fm C
1 3 5 | 1 7 1 . | 1 }

Hasil generator akord ditampilkan pada Gambar 4.



Gambar 4. Tampilan hasil dengan isi dari *list* yang dipilih

HASIL ANALISA SISTEM

Dalam menghasilkan akord, aplikasi ini mengadaptasi logika perkalian langsung, sehingga aplikasi ini membutuhkan memori yang cukup besar ketika dijalankan. Oleh karena permasalahan memori tersebut, aplikasi ini memiliki keterbatasan dalam menghasilkan variasi susunan akord. Jika susunan akord yang dihasilkan lebih dari 607.500 susunan akord, maka program akan *error* karena kehabisan memori.

KESIMPULAN

Pengadaptasian alogritma *direct product* ke dalam proses generator akord berhasil diimplementasikan dengan jumlah variasi susunan akord yang terbatas. Jumlah dari variasi susunan akord tergantung dari jumlah notasi acuan dari *string input* dan jumlah akord pada setiap notasi acuan. Hal ini disebabkan operasi dengan *direct product* memakai *resource* memori yang cukup besar.

DAFTAR PUSTAKA

1. Corozine, V., 2002, Arranging Music for the Real World: Classical and Commercial Aspects. Pacific, MO: Mel Bay. http://en.wikipedia.org/wiki/Arrangement. Diakses tanggal 1 Mei 2009.
2. Yoshioka, T., Kitahara, T., Komatani, K., Ogata, T., Okuno, H. G., 2004, Automatic Chord Transcription With Concurrent Recognition Of Chord Symbols And Boundaries, In: *Proceedings of the 5th International Conference on Music Information Retrieval ISMIR 2004*, Barcelona, Spain (2004) , p. 100-105.
3. Paiement, J.F., Eck, D., Bengio, S., 2005, A Probabilistic Model for Chord Progressions, In: *Proceedings of the 6th International Conference on Music Information Retrieval ISMIR 2005*, London: University of London: (2005), p. 312-319.
4. Chemillier, M., 2001, Improvising Jazz Chord Sequences by Means of Formal Grammars, http://ehess.modelisationsavoirs.fr/marc/publi/jim2001/icmc2001.pdf. Diakses tanggal 15 Desember 2008.
5. Mustakini, J.H., 2006, Analisis dan Desain Sistem Informasi: Pendekatan Terstruktur Teori dan Praktek Aplikasi Bisnis, Penerbit Andi, Yogyakarta.